REMARKS:

Claims 3 to 7, 10 to 17, 20 to 27 and 30 to 33 are in the application. In this paper, claims 3, 5, 7, 13, 15, 17, 23 to 25, 27 and 30 to 33 have been amended, and claims 8, 9, 18, 19, 28 and 29 have been canceled. Claims 3, 5, 7, 13, 15, 17, 23, 25, 27 and 33 are the independent claims herein. Reconsideration and further examination are respectfully requested.

## Title

The Office Action alleged that the title of the invention is not descriptive. Applicants believe that the title is descriptive. However, in order to resolve this issue, Applicants have amended the title to read "Copy on Write File System Consistency and Block Usage." This title is believed to be even more descriptive, particularly in view of the pending claims.

If the Examiner does not agree that the new title is sufficiently descriptive, then Applicants invite the Examiner to suggest a title.

## Specification

The specification was objected to for an informality. Applicants have amended the specification to correct this informality and several other minor informalities. Accordingly, withdrawal is respectfully requested of the objection to the specification.



B

Claim Objections

Claims 24 and 28 to 32 were objected to for informalities. In particular, the claims depended from the wrong base claims. Applicants have corrected this matter in the manner set forth by the Examiner, except for claims 28 and 29, which have been canceled.

Rejections Under 35 U.S.C. § 112

Claims 5, 6, 15, 16, 25 and 26 were rejected under 35 U.S.C. § 112, ¶ 2, for alleged indefiniteness. Four reasons were given for the rejections.

First, the Office Action indicated that "the specification discloses that the instant invention comprises a multitude of inode files" and that "it is unclear which inode file(s) is/are included in the characterization 'special files.'" However, while the specification does refer to plural inodes, the specification only refers to a singular "inode file" or to the "blkmap and inode files." Use of multiple inode files is not specifically disclosed. Thus, the first issue raised in the Office Action is believed to be moot.

Applicants do note that nothing in the invention precludes arrangement of the inodes in multiple inode files. In that case, it would be apparent to one skilled in the art that the "special files" would include these multiple inode files in order to contain the same information as the disclosed single inode file.

Second, the Office Action indicated that "the Examiner can find no disclosure in the specification as to what the term 'regular file' comprises." However, Applicants respectfully submit that a skilled artisan implementing the invention would recognize the plain meaning of



B

this term to be the regular types of files that are stored on a storage system. For example, such files could include, but are not limited to, some or all of data files, multimedia files, program application files, operating system definition, driver and program files, and other types of files not included in the "special files."

Third, the Office Action indicated that the limitation of requeuing dirty inodes is indefinite in the absence of a recitation of queuing dirty inodes. In response, Applicants have added a queuing step to claims 5, 15 and 25.

Fourth, the Office Action indicated that "dirty inodes" lacked antecedent basis in claims 5, 15 and 15. Applicants believe that the claims as amended now provide proper antecedent basis for this term.

In view of the foregoing, withdrawal is respectfully requested of the § 112, ¶ 2, rejections of claims 5, 6, 15, 16, 25 and 26.

Rejections Under 35 U.S.C. § 103(a)

Claims 3, 4, 13, 14, 23 and 24: These claims were rejected under 35 U.S.C. § 103(a) over "The Design of the UNIX Operating System" (Bach) in view of "The Episode File System" (Chutani) and U.S. Patent No. 5,623,666 (Pike).

Claim 3 recites a method for recording a plurality of data about a plurality of blocks of data stored in a storage system. The method includes the step of maintaining multiple usage bits for each of the plurality of blocks. One bit of the multiple usage bits for each of the plurality of blocks indicates a block's membership in an active file system and plural bits of the



B

multiple usage bits for each of the plurality of blocks indicate membership in plural read-only copies of a file system. The method also includes the step of storing, in the storage system, the multiple usage bits for each of the plurality of blocks.

The applied art, alone or in combination, is not believed to disclose or to suggest the foregoing features of claim 3, at least with respect to plural bits of the multiple usage bits for each of the plurality of blocks indicating membership in plural read-only copies of a file system.

In the Office Action, Pike was cited for teaching a "storage system ... capable of storing multiple read-only copies of a file system (see col. 6, line 48, through col. 7, line 5)." However, the cited portion states the following: "Thus, the file tree is split in two: a read-only version representing the system at the time of the dump, and an ordinary system that continues to provide normal services" (col. 6, lines 56 to 59). This line clearly indicates to Applicants that Pike's system has only one read-only version of the file system, not plural read-only copies.

Without plural read-only copies, Pike has no need for plural bits for each of a plurality of blocks to indicate membership in plural read-only copies of a file system, as recited by claim 3. Nothing in the remaining applied art is seen by Applicants to remedy Pike's deficiency in this regard. Accordingly, claim 3 and claim 4, which depends from claim 3, are believed to be allowable over the applied art. Such action is respectfully requested.

Claims 13 and 14 recite memories storing instructions that implement the methods of claims 3 and 4. Likewise, claims 23 and 24 recite systems that implement the methods of claims 3 and 4. Accordingly, claims 13, 14, 23 and 24 also are believed to be allowable over the applied art, and such action is respectfully requested.



B

Claims 5, 6, 15, 16, 25 and 26: While these claims were rejected for alleged indefiniteness under § 112, ¶ 2, no art was applied against the claims under § 103(a). Applicants have addressed the indefiniteness rejections *supra*. Accordingly, these claims are believed to be allowable, and such action is respectfully requested.

Claims 7 to 12, 17 to 22 and 27 to 33: Claims 7 to 10, 17 to 20, 27 to 30 and 33 were rejected under § 103(a) over Bach in view of Chutani. Claims 11, 21 and 31 were rejected under § 103(a) over Bach in view of Chutani and U.S. Patent No. 5,701,480 (Raz). Claims 12, 22 and 32 were rejected under § 103(a) over Bach in view of Chutani and Applicants' allegedly admitted prior art.

Claim 7 recites a method of maintaining data in a storage system. The method includes the step of maintaining a root node and inodes for a file system, with the root node pointing directly or indirectly to the inodes, and with each inode storing file data, pointing to one or more blocks in the storage system that store file data, or pointing to other inodes. The method also includes the steps of maintaining an inode map and a block map for the file system, and after data in the file system is changed, temporarily storing new data and inodes affected by the new data in memory before writing the new data and inodes affected by the new data to the storage system. The method uses a list of dirty inodes to coordinate writing the new data and inodes affected by the new data to new blocks in the storage system. The method further includes maintaining old data in old blocks in the storage system, updating the inodes and inode map to reflect the new blocks, and updating the block map, with the block map showing that both the new blocks and the old blocks are in use. By virtue of these steps, a record of changes to the file



B

system is automatically maintained in the storage system.

The applied art, alone or in combination, is not believed to disclose or to suggest the foregoing features of claim 7, at least with respect to using a list of dirty inodes to coordinate writing new data and inodes affected by the new data to new blocks in a storage system.

The Office Action cited Chutani's discussion of a log container in paragraph 4 of page 46 as teaching this feature. Applicants have carefully studied this discussion and see nothing in it that suggests that the log container is used to coordinate writing new data and inodes affected by the new data. Rather, Chutani simply states that "all meta-data updates are recorded in this log." Accordingly, Applicants respectfully submit that Chutani fails to disclose or to suggest using a list of dirty inodes to coordinate writing new data and inodes affected by the new data to new blocks in a storage system, as recited by claim 7.

Nothing in the remaining applied art is seen by Applicants to remedy Chutani's deficiency in this regard. Accordingly, claim 7 and the claims that depend therefrom are believed to be allowable over the applied art. Such action is respectfully requested.

Claim 17 recites a memory storing instructions that implement the method of claim 7. Likewise, claim 27 recites a system that implements the method of claim 7. Accordingly, claims 17 and 27 and the claims that depend therefrom also are believed to be allowable over the applied art. Such action is respectfully requested.

Closing

In view of the foregoing amendments and remarks, the entire application is believed to be in condition for allowance, and such action is respectfully requested at the Examiner's earliest convenience.

Applicants' undersigned attorney can be reached at (614) 486-3585. All correspondence should continue to be directed to the address indicated below.

Respectfully submitted,

Dane C. Butzer

Dated: January 16, 2003

Dane C. Butzer
Reg. No. 43,521

The Swernofsky Law Group
P.O. Box 390013
Mountain View, CA 94039-0013
(650) 947-0700

Changes to Specification

Pursuant to 37 C.F.R. § 1.121(b)(iii), changes to the specification effected by the accompanying paper are indicated below.

The paragraph at page 5, lines 11 to 16, has been amended as follows:

As illustrated in Figure 1, every direct pointer 110A, 112A-112B, 120A, and 122A and indirect pointer 110B and 120B in the Episode file system contains a COW bit. Blocks that have not been modified since the clone was created are contained in both the active file system and the clone, and have set (1) COW bits. The COW bit is cleared (0) [(0}] when a block that is referenced to by the pointer has been modified and, therefore, is part of the active file system but not the clone.

The paragraph at page 7, lines 9 to 22, has been amended as follows:

Creating a clone in the prior art can use up as much as 32 MB on a 1 GB disk. The prior art uses 256 MB of disk space on a 1 GB disk (for 4 KB blocks) to keep eight clones of the file system. Thus, the prior art cannot use large numbers of clones to prevent loss of data. Instead it used to facilitate backup of the file system onto an auxiliary storage means other than the disk drive, such as a tape backup device. Clones are used to backup a file system in a



B

consistent state at the instant the clone is made. By doping the file system, the clone can be backed up to the auxiliary storage means without shutting down the active file system, and thereby preventing users from using the file system. Thus, clones allow users to continue accessing an active file system while the file system, in a consistent state, is backed up. Then the clone [done] is deleted once the backup is completed. Episode is not capable of supporting multiple clones since each pointer has only one COW bit. A single COW bit is not able to distinguish more than one clone. For more than one clone, there is no second COW bit that can be set.

The paragraph at page 21, lines 4 to 11, has been amended as follows:

For a file size greater than 64 MB, the 16 buffer pointers of the inode reference double-indirect [doub!e-indirect] WAFL buffers. Each 4 KB double-indirect WAFL buffer comprises 1024 pointers pointing to corresponding single-indirect WAFL buffers. In turn, each single-indirect WAFL buffer comprises 1024 pointers that point to 4 KB direct WAFL buffers. Thus, up to 64 GB can be addressed. Figure 9D is a diagram illustrating a Level 3 inode 820 comprising 16 pointers 820B wherein pointers PTR0, PTR1, and PTR15 reference double-indirect WAFL buffers 970A, 970B, and 970C, respectively. Double-indirect WAFL buffer 970A comprises 1024 pointers that point to 1024 single-indirect WAFL buffers 980A-980B. Each single-indirect WAFL buffer 980A-980B, in turn, references 1024 direct WAFL buffers. As shown in Figure 9D, single-indirect WAFL buffer 980A references 1024 direct WAFL



B

buffers 990A-990C and single-indirect WAFL buffer 980B references 1024 direct WAFL buffers 990D-990F.

The paragraph at page 23, line 9, to page 24, line 3, has been amended as follows:

A first met-data file is the "inode file" that contains inodes describing [,] all other files in the file system. Figure 12 is a diagram illustrating an inode file 1210. The inode file 1210 may be written anywhere on a disk unlike prior art systems that write "inode tables" to a fixed location on disk. The inode file 1210 contains an inode 1210A-1210F for each file in the file system except for the inode file 1210 itself. The inode file 1210 is pointed to by an inode referred to as the "root inode". The root inode is kept in a fixed location on disk referred to as the file system information (fsinfo) block described below. The inode file 1210 itself is stored in 4 KB blocks on disk (or 4 KB buffers in memory). Figure 12 illustrates that inodes 1210A-1210C are stored in a 4 KB buffer 1220. For on-disk inode sizes of 128 bytes, a 4 KB buffer (or block) comprises 32 inodes. The incore inode file 1210 is composed of WAFL buffers 1220. When an incore inode (i.e., 820) is loaded, the on-disk inode part of the incore inode 820 is copied from the buffer 1220 of the inode file [!lie] 1210. The buffer data itself is loaded from disk. Writing data to disk is done in the reverse order. The incore inode 820, which contains a copy of the ondisk inode, is copied to the corresponding buffer 1220 of the inode file 1210. Then, the inode file 1210 is written to disk.



B

The paragraph at page 51, lines 4 to 12, has been amended as follows:

Referring to Figure 22, two previous snapshots 2110A and 2110B exist on disk. At the instant when a third snapshot is created, the root inode pointing to the active file system is copied into the inode entry 2110C for the third snapshot in the inode file 2110. At the same time in the consistency point that goes through, a flag indicates that snapshot 3 has been created. The entire file system is processed by checking if BIT0 for each entry in the blkmap fife is set (1) or cleared (0) [(0}]. All the BIT0 values for each blkmap entry are copied into the plane for snapshot three. When completed, every active block 2110-2116 and 1207 in the file system is in the snapshot at the instant it is taken.

Changes to Claims

Pursuant to 37 C.F.R. § 1.121(c)(ii), changes to any claims effected by the accompanying paper are indicated below.

Claims 3, 5, 7, 13, 15, 17, 23 to 25, 27 and 30 to 33 have been amended as follows:

3. (Amended) A method for recording a plurality of data about a plurality of blocks of data stored in a storage system, comprising the steps of:

maintaining multiple usage bits for each of said plurality of blocks, wherein one bit of said multiple usage bits for each of said plurality of blocks indicates a block's membership in an active file system and plural [one or more] bits of said multiple usage bits for each of said plurality of blocks indicate membership in plural [one or more] read-only copies of a file system; and

storing, in said storage system, said multiple usage bits for each of said plurality of blocks.

5. (Amended) A method for generating a consistency point for a storage system, comprising the steps of:



B

marking a plurality of inodes pointing to a plurality of modified blocks in a file system stored on said storage system as being in a consistency point;

flushing regular files to said storage system;

flushing special files to said storage system;

flushing at least one block of file system information to said storage system;

queuing dirty inodes after said step of marking and before said step of flushing at least one block of file system information; and

requeuing [requeueing] any of said dirty inodes that were not part of said consistency point after said step of flushing at least one block of file system information.

7. (Amended) A method of maintaining data in a storage system, comprising the steps of:

maintaining a root node and inodes for a file system, the root node pointing directly or indirectly to the inodes, and each inode storing file data, pointing to one or more blocks in the storage system that store file data, or pointing to other inodes;

maintaining an inode map and a block map for the file system; and

after data in the file system is changed, temporarily storing new data and inodes affected by the new data in memory before writing the new data and inodes affected by the new data to the storage system, using a list of dirty inodes to coordinate writing the new data and inodes affected by the new data to new blocks in the storage system [writing new data to one or more new blocks in the storage system], maintaining old data in old blocks in the storage system,



B

updating the inodes and inode map to reflect the new blocks, and updating the block map, with the block map showing that both the new blocks and the old blocks are in use;

whereby a record of changes to the file system is automatically maintained in the storage system.

13. (Amended) A memory storing information including instructions, the instructions executable by a processor to record a plurality of data about a plurality of blocks of data stored in a storage system, the instructions comprising the steps of:

maintaining multiple usage bits for each of said plurality of blocks, wherein one bit of said multiple usage bits for each of said plurality of blocks indicates a block's membership in an active file system and plural [one or more] bits of said multiple usage bits for each of said plurality of blocks indicate membership in plural [one or more] read-only copies of a file system; and

storing, in said storage system, said multiple usage bits for each of said plurality of blocks.

15. (Amended) A memory storing information including instructions, the instructions executable by a processor to generate a consistency point for a storage system, the instructions comprising the steps of:

marking a plurality of inodes pointing to a plurality of modified blocks in a file system stored on said storage system as being in a consistency point;



B

flushing regular files to said storage system;

flushing special files to said storage system;

flushing at least one block of file system information to said storage system;

queuing dirty inodes after said step of marking and before said step of flushing at least one block of file system information; and

requeuing [requeueing] any of said dirty inodes that were not part of said consistency point after said step of flushing at least one block of file system information.

17. (Amended) A memory storing information including instructions, the instructions executable by a processor to maintain data in a storage system, the instructions comprising the steps of:

maintaining a root node and inodes for a file system, the root node pointing directly or indirectly to the inodes, and each inode storing file data, pointing to one or more blocks in the storage system that store file data, or pointing to other inodes;

maintaining an inode map and a block map for the file system; and

after data in the file system is changed, temporarily storing new data and inodes affected by the new data in memory before writing the new data and inodes affected by the new data to the storage system, using a list of dirty inodes to coordinate writing the new data and inodes affected by the new data to new blocks in the storage system [writing new data to one or more new blocks in the storage system], maintaining old data in old blocks in the storage system,



B

updating the inodes and inode map to reflect the new blocks, and updating the block map, with the block map showing that both the new blocks and the old blocks are in use;

whereby a record of changes to the file system is automatically maintained in the storage system.

23. (Amended) A system comprising:

a processor;

a storage system; and

a memory storing information including instructions, the instructions executable by the processor to record a plurality of data about a plurality of blocks of data stored in the storage system, the instructions comprising the steps of: (a) maintaining multiple usage bits for each of said plurality of blocks, wherein one bit of said multiple usage bits for each of said plurality of blocks indicates a block's membership in an active file system and plural [one or more] bits of said multiple usage bits for each of said plurality of blocks indicate membership in plural [one or more] read-only copies of a file system; and (b) storing, in said storage system, said multiple usage bits for each of said plurality of blocks.

24. (Amended) A system as in claim 23 [13], wherein one or more bits of said multiple usage bits for each of said plurality of blocks further indicate block reusability.

25. (Amended) A system comprising:



B

a processor;

a storage system; and

a memory storing information including instructions, the instructions executable by the processor to generate a consistency point for the storage system, the instructions comprising the steps of: (a) marking a plurality of inodes pointing to a plurality of modified blocks in a file system stored on said storage system as being in a consistency point; (b) flushing regular files to said storage system; (c) flushing special files to said storage system; (d) flushing at least one block of file system information to said storage system; (e) queuing dirty inodes after said step of marking and before said step of flushing at least one block of file system information; and (f) [(e)] requeuing [requeueing] any of said dirty inodes that were not part of said consistency point after said step of flushing at least one block of file system information.

27. (Amended) A system comprising:

a processor;

a storage system; and

a memory storing information including instructions, the instructions executable by the processor to maintain data in the storage system, the instructions comprising the steps of: (a) maintaining a root node and inodes for a file system, the root node pointing directly or indirectly to the inodes, and each inode storing file data, pointing to one or more blocks in the storage system that store file data, or pointing to other inodes; (b) maintaining an inode map and a block map for the file system; and (c) after data in the file system is changed, temporarily



B

storing new data and inodes affected by the new data in memory before writing the new data and inodes affected by the new data to the storage system, using a list of dirty inodes to coordinate writing the new data and inodes affected by the new data to new blocks in the storage system [writing new data to one or more new blocks in the storage system], maintaining old data in old blocks in the storage system, updating the inodes and inode map to reflect the new blocks, and updating the block map, with the block map showing that both the new blocks and the old blocks are in use;

whereby a record of changes to the file system is automatically maintained in the storage system.

30. (Amended) A system as in claim 27 [17], wherein the instructions further comprise the step of creating a snapshot of the file system by copying the root node.

31. (Amended) A system as in claim 30 [20], wherein the block map indicates membership of blocks in one or more snapshots.

32. (Amended) A system as in claim 30 [20], wherein the instructions further comprise the step of deleting a snapshot from the storage system, wherein blocks that are only part of the deleted snapshot are released for re-use by the storage system.



B

33. (Amended) A system for maintaining data in storage means, comprising the steps of:

means for maintaining a root node and inodes for a file system, the root node pointing to inodes, and each inode storing file data, pointing to one or more blocks in the storage means that store file data, or pointing to other inodes;

means for maintaining an inode map and a block map for the file system; and

means for, after data in the file system is changed, temporarily storing new data and inodes affected by the new data in memory before writing the new data and inodes affected by the new data to the storage system, using a list of dirty inodes to coordinate writing the new data and inodes affected by the new data to new blocks in the storage system [writing new data to one or more new blocks in the storage system], maintaining old data in old blocks in the storage system, updating the inodes and inode map to reflect the new blocks, and updating the block map, with the block map showing that both the new blocks and the old blocks are in use;

whereby a record of changes to the file system is automatically maintained in the storage means.

Claims 8, 9, 18, 19, 28 and 29 have been canceled.



B